Probabilistic decision graphs for optimization under uncertainty

Jensen, Finn V.; Nielsen, Thomas Dyhre

Published in:
4OR

DOI (link to publication from Publisher):
10.1007/s10288-011-0159-7

Publication date:
2011

Document Version
Early version, also known as pre-print

Noname manuscript No.
(will be inserted by the editor)

Probabilistic Decision Graphs for Optimization under Uncertainty

Finn V. Jensen · Thomas Dyhre Nielsen

Received: date / Accepted: date

Abstract This paper provides a survey on probabilistic decision graphs for modeling and solving decision problems under uncertainty. We give an introduction to influence diagrams, which is a popular framework for representing and solving sequential decision problems with a single decision maker. As the methods for solving influence diagrams can scale rather badly in the length of the decision sequence, we present a couple of approaches for calculating approximate solutions.

The modeling scope of the influence diagram is limited to so-called symmetric decision problems. This limitation has motivated the development of alternative representation languages, which enlarge the class of decision problems that can be modeled efficiently. We present some of these alternative frameworks and demonstrate their expressibility using several examples. Finally, we provide a list of software systems that implement the frameworks described in the paper.

Keywords Survey, probabilistic decision graphs, influence diagrams.

1 Introduction

Bayesian networks (Pearl, 1988) have for a couple of decades been a popular framework for modeling and reasoning under uncertainty. Bayesian networks have also been exploited in the development of frameworks for decision making under uncertainty. These types of frameworks are commonly called *probabilistic decision graphs*. Most prominently is the influence diagram (Howard and Matheson, 1981), which originally was proposed as a compact representation of symmetric decision trees.

The current paper provides an overview on probabilistic decision graphs. The concept of Bayesian networks is briefly introduced, but hopefully sufficiently for

Finn V. Jensen and Thomas Dyhre Nielsen
Department of Computer Science
Aalborg University
Denmark
E-mail: {fvj,tdn}@cs.aau.dk

reading the paper. However, readers unfamiliar with Bayesian networks are recommended to consult a modern textbook like (Jensen and Nielsen, 2007), (Korb and Nicholson, 2004), (Kjaerulff and Madsen, 2008), (Darwiche, 2009), and (Koller and Friedman, 2009).

1.1 Notation

A *variable* has a set of mutually exclusive and exhaustive *states*. Variables are denoted by upper case letters; a state of a variable is denoted by a lower case letter; a set of variables is denoted by an upper case boldface letter, and a configuration of states over a set $\mathbf{X}$ is denoted by $\mathbf{x}$, a lower case boldface letter. In this paper all variables have only a finite set of states.

A conditional probability, "the probability of A being in state a given that B is in state b" is denoted $P(A = a \,|\, B = b)$, and also $P(a \,|\, b)$ when the variables are obvious from the context. $P(A \,|\, B)$ denotes a table of conditional probabilities $P(a \,|\, b)$, one for each combination of states of A and B, and $P(\mathbf{X} \,|\, \mathbf{Y})$ denotes a table of conditional probabilities $P(\mathbf{x} \,|\, \mathbf{y})$.

2 Normative decision making

In normative decision making you model a domain using probabilities and utilities, and you aim at maximizing the expected utility. For a single decision, it may be illustrated as in Figure 1.

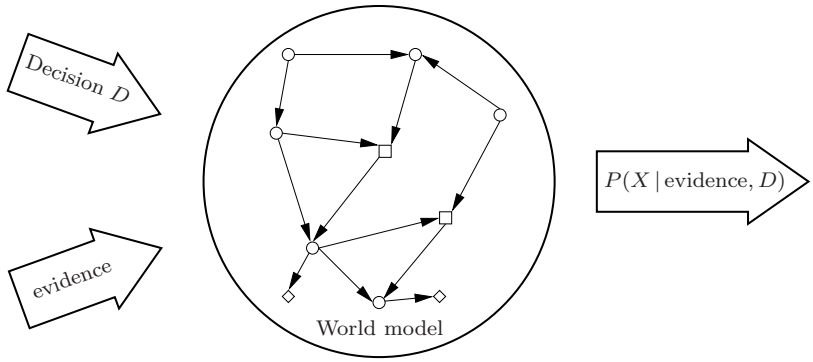

Fig. 1 A general model for one decision.

You have a *decision variable* D consisting of a set of options, and you have a model of the world relevant for D. The world model consists of a finite set $\mathbf{W}$ of variables and the relations between them. Furthermore, you have a *utility function* $U(\mathbf{X}, D)$, where $\mathbf{X} \subseteq \mathbf{W}$. For a particular decision problem you have a set $\mathbf{o}$ of *observations* of a set of variables $\mathbf{O} \subseteq \mathbf{W}$. The world model shall provide $P(\mathbf{X} \,|\, \mathbf{o}, D)$ s.t. you can calculate *the expected utility of D given $\boldsymbol{o}$*:

$$EU(D \mid \mathbf{o}) = \sum_{\mathbf{X}} U(\mathbf{X}, D) P(\mathbf{X} \mid \mathbf{o}, D). \tag{1}$$

Following the normative approach, you are assumed to act *rationally*, which means that you choose the option of maximal expected utility (von Neumann and Morgenstern, 1944):

$$Opt(D \mid \mathbf{o}) = \arg\max_{D} \sum_{\mathbf{X}} U(\mathbf{X}, D) P(\mathbf{X} \mid \mathbf{o}, D). \tag{2}$$

The function, which for each configuration of $\mathbf{O}$ returns an optimal decision is denoted $\delta_D(\mathbf{O})$ and is called an *optimal policy for D*.

2.1 Example: a simplified poker game

You have entered a poker game with the following rules. There are two players, Opponent O and you M, and each player places 1€ when entering the game. There are two rounds of card change, where O changes first each time. After the card change rounds, O has to decide whether to *call*, which costs 1€, or to *fold*. If O calls, M has to decide whether to *fold* (in which case O takes the pot) or to *call*. Calling costs 1€, and if M calls, the players compare hands; the one with the best hand takes the pot.

Now, M has been through the card changes, O has called, and M has to decide whether to call or fold. The relevant information available is M's own hand, O's change of cards, and the fact that she called. M can use this information to estimate O's hand OH. In Figure 2, a model for this task is presented.

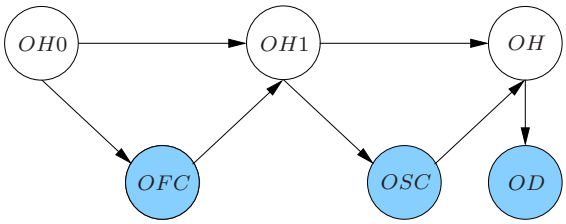

Fig. 2 A Bayesian network for calculating $P(OH \mid ofc, osc, od)$.

The model is a *Bayesian network* (Pearl, 1988). The white nodes represent the opponent's initial hand ($OH0$, a variable with states describing the relevant types of initial hands), her hand after the first change of cards ($OH1$), and her final hand (OH). The colored nodes represent variables whose states M knows when taking the decision of calling or folding. OFC represents the number of cards in O's first change of cards, OSC represents her second change of cards, and OD represents her decision of calling or folding. The links between nodes represent causal impact. That is, $OH0$ and OFC have a causal impact on her second hand. As the impact is not deterministic, it is represented by conditional probabilities. For example, the node $OH1$ has attached the conditional probability table $P(OH1 \mid OH0, OFC)$

and the node $OH0$ has a prior distribution over the possible hands attached.[1] The Bayesian network in Figure 2 can be used to calculate $P(OH \mid ofc, osc, od)$, and Equation 1 can be used to calculate the expected utility of calling with the possible monetary wins and losses as the utilities of the game.

Actually, the model in Figure 2 can be extended to also represent M's decision problem (see Figure 3, which is no longer a Bayesian network).

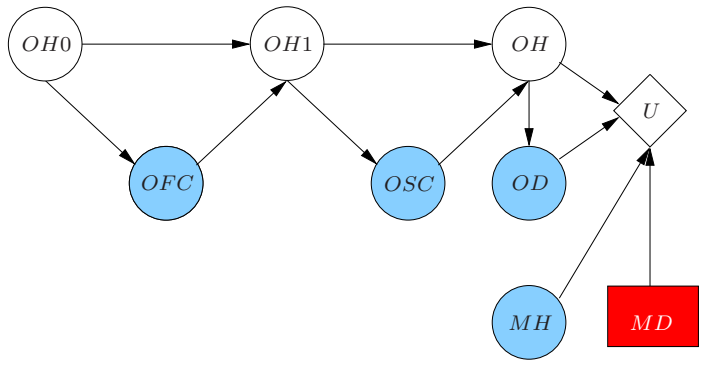

Fig. 3 A graphical model representing M's problem of whether to call or fold.

In Figure 3 the diamond shaped node represents M's utilities for the various scenarios (the amount of €s to win or loose), and the rectangular shaped node represents M's options (call of fold).

Now, if the states of the variables are specified as well as the conditional probability tables and the utility function, then sufficient information has been provided for solving the decision problem. In other words, it is sufficient to enter the graph, the probability tables and the utility function to the computer; then it can take care of the calculations.

2.2 Bayesian networks

Definition 1 A *Bayesian network* consists of the following:

- A set of *variables* and a set of *directed edges* between variables.
- Each variable has a finite set of mutually exclusive and exhaustive states.
- The variables together with the directed edges form a directed acyclic graph (a DAG).
- To each variable A with parents $\mathrm{pa}(A)$, there is a conditional probability table $P(A \mid \mathrm{pa}(A))$ attached.

Note that if A has no parents, then the table reduces to the unconditional (prior) probability table $P(A)$.

The backbone for the application of Bayesian networks is the following theorem (Pearl, 1988).

[1] The conditional probability tables associated with OFC, OSC, and OD are a little special; they encode the policy which M believes that O follows when deciding on the change of cards and whether to call of fold. We shall return to this issue in Section 7.

Theorem 1 (The chain rule for Bayesian networks) *Let BN be a Bayesian network over $\boldsymbol{U} = \{A_1, \ldots, A_n\}$. Then BN specifies a unique joint probability distribution $P(\boldsymbol{U})$ given by the product of all the conditional probability tables specified in BN:*

$$P(\boldsymbol{U}) = \prod_{i=1}^{n} P(A_i \mid \mathrm{pa}(A_i)).$$

So, a Bayesian network is a compact representation of a joint probability distribution, but it is also a graphical model representing cause-effect-relations of a domain and reflecting the inherent uncertainty in the domain. It should be stated here that probability theorists are not religious about the links being causal as long as the model represents the proper distribution. However, when you include decisions which have an impact on the state of some variables, then you should be aware that the effect of an action follows the direction of causal links, and in these situations it is critical that the direction actually represents a causal direction.

The primary application of Bayesian networks is *belief updating*. Confronted with a particular case you will receive case specific knowledge. We shall by *evidence* mean a statement that a particular variable is in a particular state. In that case we say that the variable is *instantiated*. Belief updating consists of entering the evidence to the Bayesian network model and calculating the posterior probability distribution for other sets of variables.

In general, the belief updating task is exponential in the number of variables. However, good exact and approximate algorithms have been constructed, such that belief updating is tractable for most of the Bayesian networks you meet. We shall not deal with belief updating algorithms in this paper, for which the reader is referred to the textbooks on Bayesian networks listed in Section 1.

3 A sequence of decisions

When you have a sequence of decisions, the modeling becomes more involved. To find an optimal first decision, you have to anticipate the future. That is, you have to take all future scenarios into account, and you must forecast your future decisions. The model for your first decision can therefore be seen as a nested model with internal models for the future decisions (see Figure 4).

The model in Figure 4 contains a sequence of three decisions. To take the first decision, you consider the scenario for the next decision: what will be the information at hand, and what will the decision be? Assume that you know that the state of the variables $\mathbf{X}$ will be disclosed before you take the next decision. However, you do not yet know the actual states, so you have to take all possible configurations of $\mathbf{X}$ into account and calculate the probabilities of these configurations. You may then use the model to calculate the expected utility of the various options given the evidence. If you assume that you in the future will act rationally, you can establish a policy function $\delta_{\mathbf{X}}$, which provides an optimal decision given the evidence.

However, you cannot determine the expected utility for the options of the second decision without considering the third decision. Therefore, you start by solving the decision problem for the last decision. When taking the last decision, you will know the state of the variables $\mathbf{Y}$, and the solution (found using the

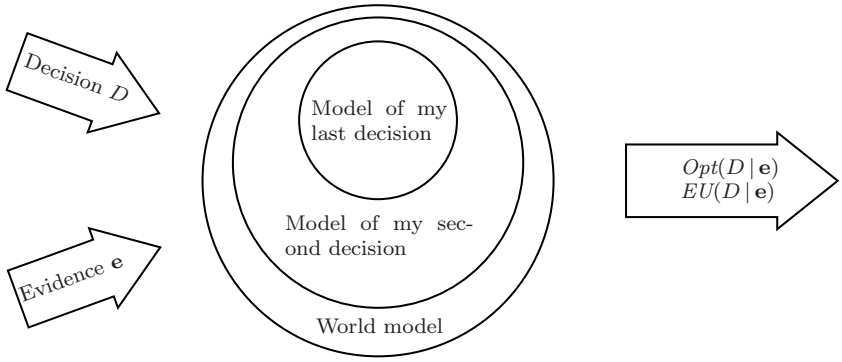

Fig. 4 A model for my first decision containing models for two future decisions.

approach outlined in Section 2) provides a policy function, δ_{last}, which for evidence $\mathbf{y}$ gives the optimal decision $\delta_{\text{last}}(\mathbf{y})$. The policy function for the last decision can now be used in the model for the second last decision, and in this way you work your way backwards to the first decision.

3.1 Poker again

In the poker game you have in fact three decisions to take, namely two changes of cards and the final decision on whether to call or fold. Following the reasoning above, we start with the last decision (see Figure 5).

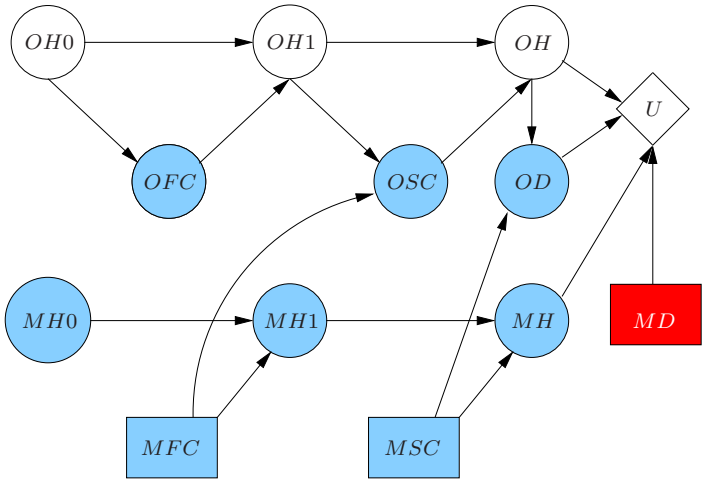

Fig. 5 A model for the last decision including the preceding card changes.

The model in Figure 5 reflects that the opponent observes your card changes. The nodes MFC and MSC represent M's first and second change of cards, respectively. These nodes are given a rectangular shape because they, like MD, are

decision nodes. Furthermore, these nodes are not given probabilities as they are always under the full control of the decision maker.

From the model the expected utility of *call* is calculated:

$$EU(\mathit{call} \,|\, \mathbf{evidence}) = \sum_{OH} U(OH, \mathit{mh}, \mathit{od}, \mathit{call}) P(OH \,|\, \mathbf{evidence}),$$

where **evidence** is a configuration over the variables OFC, OSC, OD, $MH0$, MFC, $MH1$, MSC, and MH. The model also provides a policy $\delta_{MD}(OFC, OSC, OD, MH0, MFC, MH1, MSC, MH)$ for MD. The domain of δ_{MD} consists of the variables whose states are known at the time of deciding on MD, i.e., the variables in the past of MD. This set of variables (and thereby the domain of the policy) can become intractably large, but often not all past variables are relevant for the decision. There are algorithms for performing a structural analysis to determine the relevant past (Nielsen and Jensen, 1999; Shachter, 1999; Lauritzen and Nilsson, 2001). For the model in Figure 5, the analysis gives that $MH0$ and $MH1$ are irrelevant. You may wonder why MSC is relevant as MSC does not have a causal impact on an unknown variable in the domain of U. However, when O decides for calling or folding, she knows MSC. This, together with her own hand, determines her decision of calling. Therefore, when M knows that she called, MSC will tell him something about her hand. Assume for example, that M changed no cards. Then O may infer that he most probably has a straight or better, and with three of a kind or less, she will be inclined to fold. Now, M can do the same reasoning: as she called, she does most probably not have three of a kind or less, and he may with a small straight decide to fold. Note that for this model, the possibility of her bluffing (and her estimate of M's bluffing) is represented by the conditional probabilities of her policies.

The kind of reasoning performed above exploit the properties of causality. It is formalized in the so-called rules of *d-separation* (Pearl, 1988), which work on the model structure and can reveal conditional independence: if X and Y are d-separated given Z, then X and Y are conditionally independent given Z. The implication does, however, not hold the other way around. The interested reader is referred to standard texts on Bayesian networks (see the references in Section 1).

When $\delta_{MD}(OFC, OSC, OD, MFC, MSC, MH)$ has been determined, it is used as a conditional probability table in the model for the second card change (see Figure 6). This conditional probability table is also called a *chance variable representation of the policy* δ_{MD}.

From the model in Figure 6 we can calculate

$$EU(MSC \,|\, \mathbf{evidence}) = \sum_{OH,OD,MH,MD} U(OH, OD, MH, MD) \\ P(OH, OD, MH, MD \,|\, \mathbf{evidence}, MSC),$$

together with a policy $\delta_{MSC}(OFC, OSC, MFC, MH1)$ for MSC. Using the same techniques as above, we can get a model for the first change of cards (see Figure 7), which can finally be used to find an optimal policy for MFC.

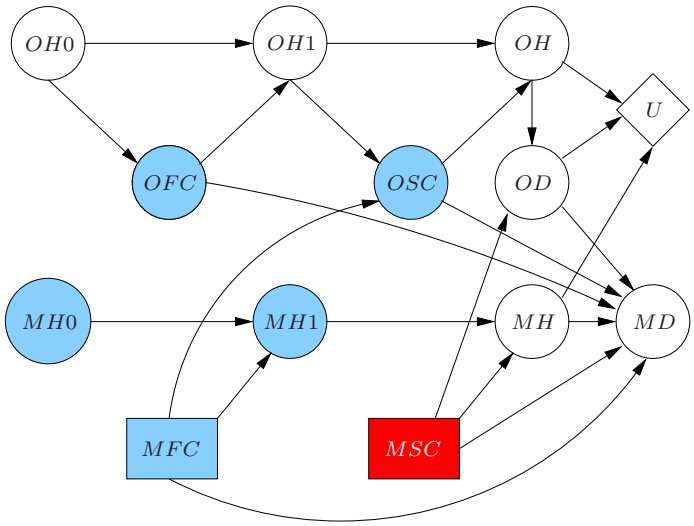

Fig. 6 A model for the second change of cards. MD is a chance node with δ_{MD} as conditional probability table.

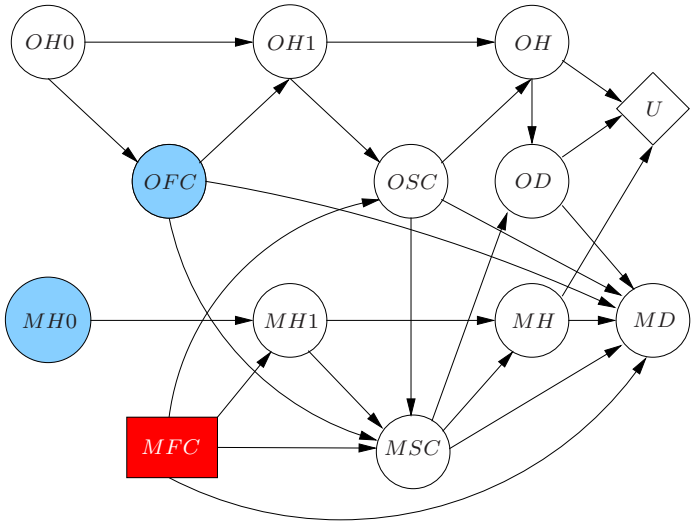

Fig. 7 A model for the first decision.

4 Influence diagrams

The decision problem above is represented by three models, one for each decision. The models have many similarities, and actually, they can be represented in a single unified model (see Figure 8).

The model in Figure 8 is called an *influence diagram* (Howard and Matheson, 1981). It contains a Bayesian network part and a decision part. The rectangular shaped nodes represent the decisions, and an edge from a node N to a decision

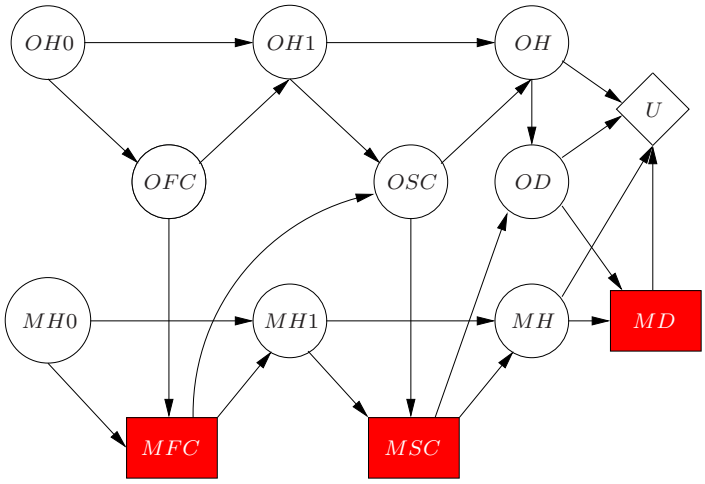

Fig. 8 An influence diagram representing the poker problem.

node D means that the state of N is known when D is to be decided. We assume *no-forgetting*: the decision maker remembers what he has known and decided in the past.

We can read a partial order of observations and decisions by using the fact that a node, which is a consequence of a decision D, cannot be observed before D is decided upon. From Figure 8 we can read the following order: $\{MH0, OFC\} \prec MFC \prec \{MH1, OSC\} \prec MSC \prec \{MH, OD\} \prec MD$. Therefore, the influence diagram in Figure 8 can be unfolded to the three models in Figures 5 to 7.

Definition 2 An *influence diagram* (*ID*) consists of a DAG over chance nodes, decision nodes, and utility nodes with the following structural properties (see Figure 9 for a general example):

- there is a directed path comprising all decision nodes;
- the utility nodes have no children, and they have no states;
- the chance nodes and the decision nodes have a finite set of mutually exclusive and exhaustive states.

Furthermore, a conditional probability table $P(A \mid \mathrm{pa}(A))$ is attached to each chance node A, and a real-valued function over $\mathrm{pa}(V)$ is attached to each utility node V.

4.1 Semantics

Edges into a decision node are called *information links*, and they indicate that the state of the parents are known prior to taking the decision. No-forgetting is assumed, so the variables known when taking decision $D4$ in Figure 9 are $\{O1, O2, D1, O3, O4, D2, O5, O6, D3, O7, O8\}$.

The structural requirement that there must a path comprising all decision nodes ensures that the influence diagram defines a temporal sequence of decisions.

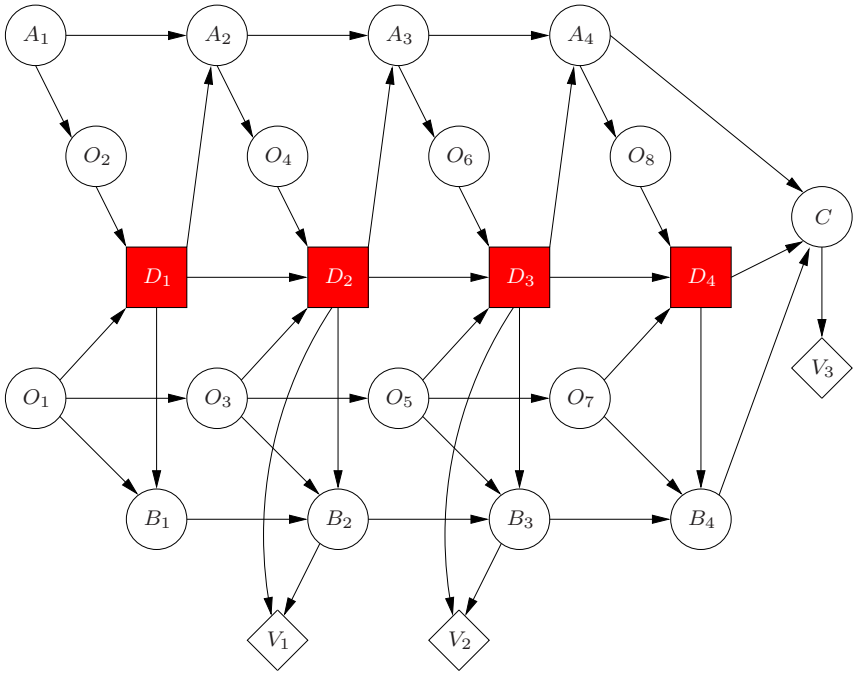

Fig. 9 An influence diagram with three utility functions and a sequence of four decisions.

This yields a partitioning of the chance variables into disjoint subsets according to the time of observation. The set $\mathcal{I}_0$ is the set of variables observed before any decision is taken ($\{O1, O2\}$ in Figure 9); $\mathcal{I}_1$ is the set of variables observed after the first decision and before the second, and so forth.

When there are several utility functions in the model, the total utility is the sum of the utility functions. This includes products of positive utilities as it can be transformed to sums of the logarithms. Other combinations of utilities can be represented by introducing super-value nodes (Luque and Díez, 2010).

4.2 Evaluating influence diagrams

A *solution* of an influence diagram is an *optimal strategy*. A strategy consists of a set of policies, one for each decision node. A policy for D is function which, for each configuration of the relevant past of D, returns a decision in D. An optimal strategy is a set of policies, which together gives the decision maker maximal expected utility. The process of determining an optimal strategy is often referred to as *evaluating* the influence diagram.

As sketched in Section 3.1, a conceptually simple method for evaluating an ID is to first find an optimal policy for the last decision node, substitute that with a chance variable representation of its policy, and then continue to the second last decision node .... Other and more efficient methods have been constructed (Shachter, 1986; Shenoy, 1992; Jensen et al, 1994), and the interested reader can find them in the textbooks (Jensen and Nielsen, 2007; Kjaerulff and Madsen, 2008).

4.3 Modeling scope

Influence diagrams are very handy for representation. As opposed to decision trees, specification of an influence diagram does not require much effort. The hard work is left to the computer. However, influence diagrams can only efficiently model *symmetric* scenarios, namely scenarios with a fixed sequence of decisions and where the variables observed between decisions are the same regardless of the previous decisions and observations. This is a rather delimiting constraint. For example, if the decision is whether or not to perform a test, then the next observation is dependent on that decision. Assume, for example that you may perform a test *Test?* (of cost C), which discloses the state of the variable O with states $o_1, \ldots o_n$, and after the decision *Test?* you shall take the decision D. You cannot make O a parent of D, as O is only observed if you decide to test.

There is a way of overcoming this problem. Consider Figure 10, where a new child O' of O is introduced. O' has the states $\{o_1, \ldots o_n, \textit{no-test}\}$, and the conditional probability table $P(O' \mid O, \textit{Test?})$ is so that if $\textit{Test?} = \textit{no}$ then $O' = \textit{no-test}$; otherwise $O' = O$.

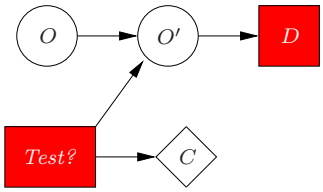

Fig. 10 A part of an influence diagram representing a test decision.

This approach works fine if there is only one test to decide upon. It is much more complex when you have several tests. Assume, for example, that you after an initial investigation I have two tests T_A and T_B for a set of diseases *Dis* and a single subsequent treatment decision D. One strategy may be to perform T_A, and depending on the outcome of the test you perform T_B, but you may also do it in the opposite order, or you may perform no test at all. Following the approach above you end up with the influence diagram in Figure 11.

For more than two tests, the influence diagram representation becomes quite inhuman. The decision nodes contain all tests as options, and the observation nodes have as states all possible outcomes of all possible tests. Furthermore, you have to make it explicit that a tests shall never be repeated. In short, the representation scales badly, and the models are hard to read.

5 Unconstrained influence diagrams

A test decision is one example of *asymmetry* in a test scenario. Much effort is now spent on constructing languages for easy representation of asymmetric decision scenarios (Shenoy, 2000; Jensen et al, 2006). In this section we present one such specification language. An *unconstrained influence diagram* (UID) (Jensen and Vomlelova, 2002) is an influence diagram, except that it is not required that there is

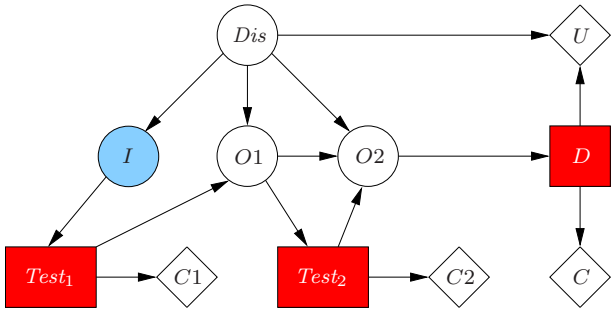

Fig. 11 An influence diagram representing two tests and one subsequent decision. The $Test$ nodes have three options: t_A, t_B, and *no-test*. The O nodes have five states: pos_A, pos_B, neg_A, neg_B, *no-test*. The edge $O1 \rightarrow O2$ models that repeating a test will give identical results.

a directed path comprising all decision nodes. Instead, nodes which are eventually observed are called *observables* and drawn as double-circled nodes (see Figure 12).

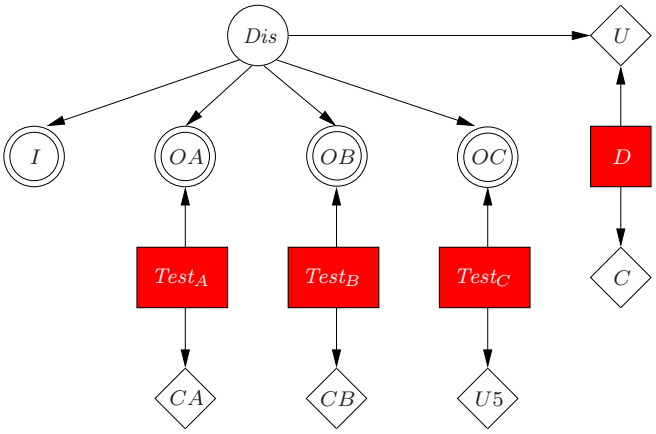

Fig. 12 An unconstrained influence diagram representing a scenario with three tests and one decision.

The test decisions as well as their outcomes are represented explicitly (a general assumption behind this modeling is that the tests are of the kind that they will give the same result if repeated). The nodes I, OA, OB, and OC are labeled as observables. An observable can be observed when all its preceding decisions have been taken. In that case, the node is said to be *released*. As the costs are part of the (test) decisions, we can safely assume that an observable is observed whenever it is released (nothing can be gained by waiting).

When the model specified in Figure 12 is extended with the required parameters (probabilities and utilities), the decision problem is fully specified, and it can be left to the computer to determine an optimal strategy. Note that a strategy for a UID differs from a strategy for an ID as it contains a specification of the decision to choose as well as a specification of which decision variable to consider next; the latter part of the strategy is also referred to as a *next-step function*. An optimal

strategy for the UID in Figure 12 could for example specify that if I is y, then perform $Test_B$ and continue to decide on $Test_A$ followed by a decision on $Test_C$; if I is n, then perform $Test_C$, and if the result is pos_C then perform $Test_A$, else perform $Test_B$, etc.

5.1 Evaluating UIDs

An optimal strategy for a UID is a set of next-step functions and a set of policies for each decision. The possible paths of decisions and observations in a strategy can be put together into a DAG. Figure 13 shows an example of a DAG for a possible optimal strategy (the one outlined above).

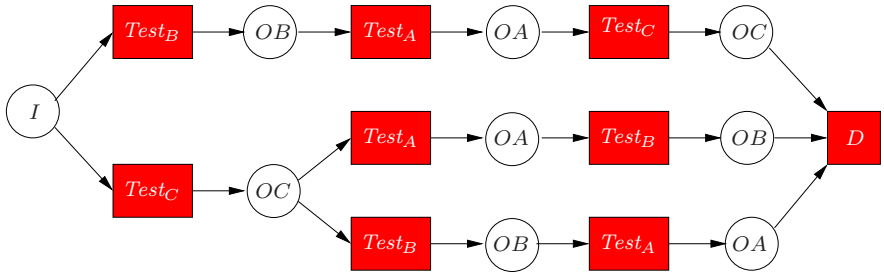

Fig. 13 A DAG for an optimal strategy for the UID in Figure 12.

The principle in evaluating a UID is to establish a DAG which includes all paths possible in an optimal strategy. Some paths may be discarded. For the UID in Figure 12 it can be inferred that an optimal path must have D as the last decision. If, namely, D has been decided, performing a test will only be a cost and should be skipped, and the decision on skipping a test can just as well be taken before the decision on D is taken. Actually, this kind of reasoning can be performed algorithmically on the basis of a set of rules for analyzing UIDs (Ahlmann-Ohlsen et al, 2009).

The result of the analysis is a so-called *GS-DAG*, which has the property, that it contains a DAG for an optimal strategy as a subgraph. A GS-DAG for the UID in Figure 12 is shown in Figure 14. Note that the DAG in Figure 13 is a subgraph of the DAG in Figure 14.

When the GS-DAG has been established, the policies are determined in the usual backwards order, where you bring the expected utilities with you. When two paths meet in the backwards solution phase, the expected utilities are used to determine the next-step functions in the (forward) strategy.

6 Representation of solutions for influence diagrams

The main complexity issue regarding influence diagrams is the size of the domains of the policies in an optimal solution. The more observations and decisions there are, the larger will the domains for the policies of the last decisions be. Consider

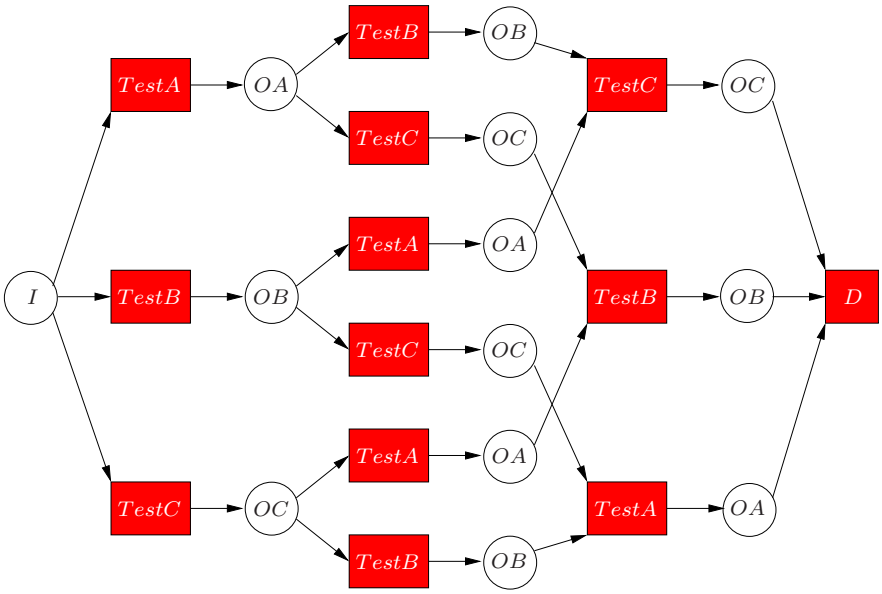

Fig. 14 A GS-DAG for the UID in Figure 12.

the influence diagram in Figure 9, and assume that all variables have ten states. The domain for the policy δ_4 is $\{O1, O2, D1, O3, O4, D2, O5, O6, D3, O7, O8\}$. It contains 10^{11} configurations, and even if you had sufficient time to calculate the policy, it would be difficult to store it as a look-up table for fast access. Therefore, you have to look for other ways of representing the policies.

For δ_4 from Figure 9 there is an efficient representation at hand, namely the influence diagram itself (See Figure 15).

When $D4$ is to be decided, you know the configuration of the evidence. You enter the evidence to the influence diagram, and the Bayesian network part of the model can easily provide $P(C \mid D, \mathbf{evidence})$. So, even though the domains for the policies of the last decisions may be very large, they are not a problem when you eventually reach that point in your decision making.

However, they cause another complexity problem. Look back at Figure 4 and consider the evaluation method described in Section 4.2. To determine the policy δ_1 we need an online efficient representation of δ_3. If the domain of δ_3 is intractably large, it is not possible to determine an optimal policy for the first decision. This is called *the curse of knowing that you shall not forget*.

6.1 Limited memory influence diagrams

A way of addressing the curse of knowing that you shall not forget is to work with approximate representations, where you assume that you in fact do forget something. It is called a *limited memory influence diagram* (LIMID) (Nilsson and Lauritzen, 2000). The approach can be used for obtaining an approximate solution by assuming that you have less information at hand in the future, and your decisions will therefore not necessarily be optimal. In the specification we cannot

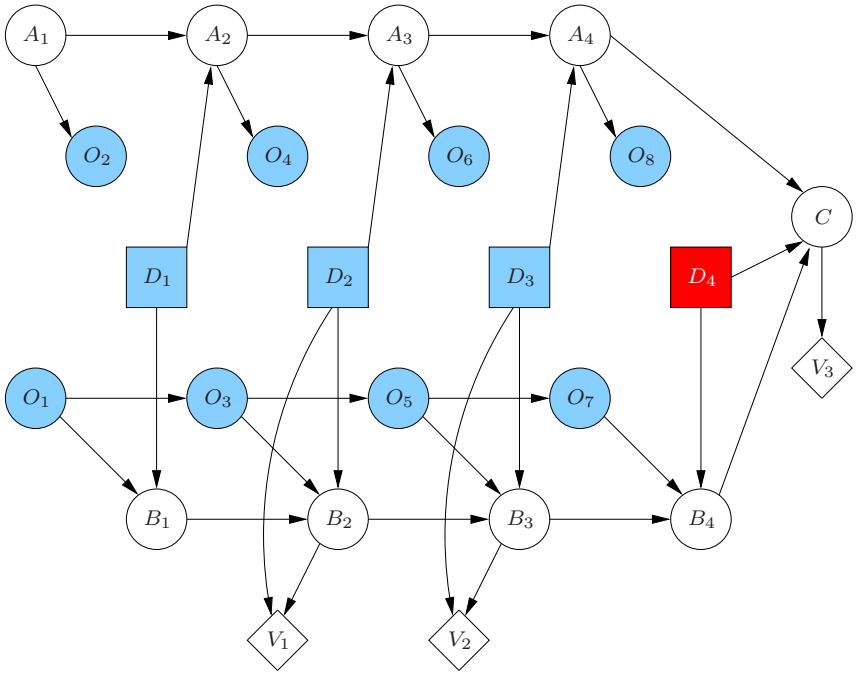

Fig. 15 A representation of the policy for the last decision.

assume no-forgetting, and all information has to be made explicit through information links. If we assume in the influence diagram in Figure 9 that we remember one decision back, we get the model in Figure 16, where the domain of δ_4 contains only five variables rather than the 11 variables for the influence diagram in Figure 9.

The policies for a LIMID are found through an iterative procedure, where you start off with a set of policies of your own choice for all but the last decision. You enter these policies as conditional probabilities and solve the resulting one-decision problem to determine δ_{last}. Then you work your way backwards as described in Section 3.1. You enter δ_{last} as a conditional probability table for the chance node D_{last}; change the second last decision to a decision node and determine an optimal policy for it; .... When you have established a policy for the first decision, you reconsider the last decision and change the policy if it will increase the expected utility; work your way backwards again, and continue the iteration until no policy has been changed. As there are only finitely many policies and the expected utility of the strategy increases through each iteration, the procedure will stop.

The result is an approximated policy for the first decision. When you get to the next decision, you have collected new information, and you can therefore do better than simply using the policy from the current LIMID; you have information at hand, which the original LIMID assumes you have forgotten. You should therefore use a new LIMID (see Figure 17) and re-run the procedure using this model to find δ_{D_2}. For example, in Figure 17 the decisions D_3 and D_4 have more relevant information at hand than in the LIMID in Figure 16.

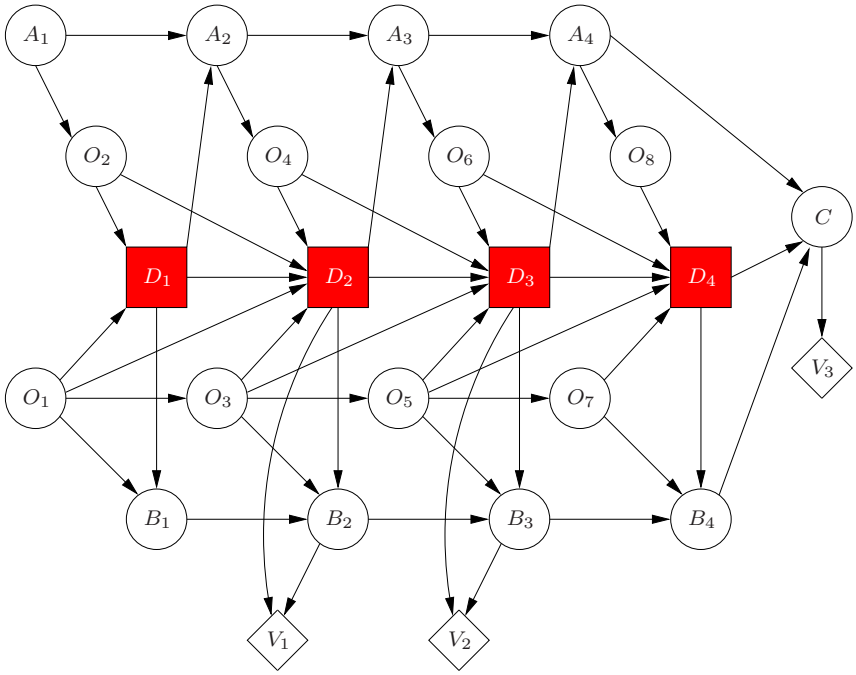

Fig. 16 A LIMID approximation of the model in Figure 9. There is no no-forgetting.

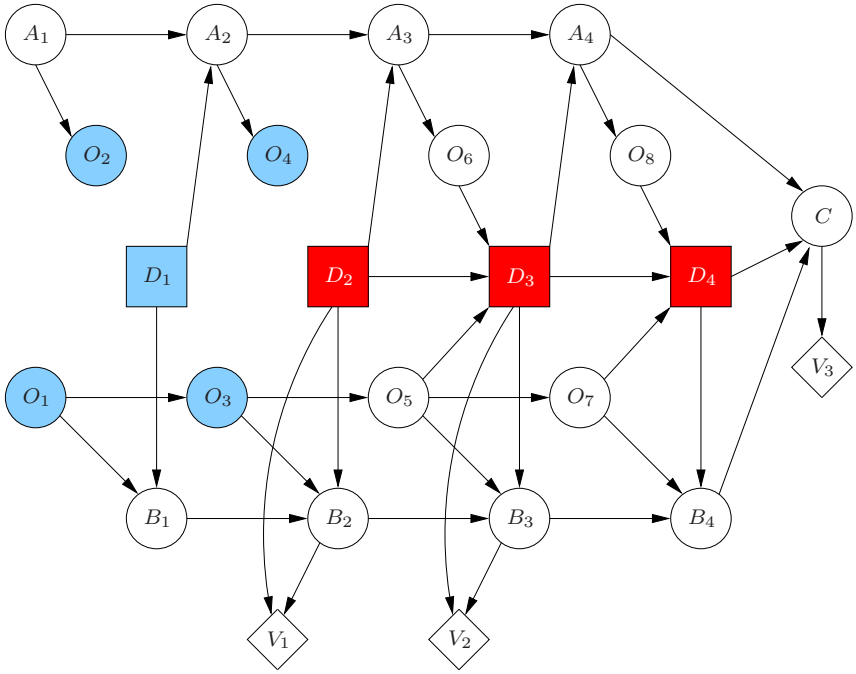

Fig. 17 A LIMID for the second decision of the model in Figure 9.

When you approach the last decisions, you can use the initial influence diagram for determining policies.

6.2 Information abstraction

A LIMID is one example of a general approach called *information abstraction*, where the information is abstracted to a less informative, but smaller size. LIMIDs are a rather simple way of abstracting information. You may use more sophisticated methods like introducing new variables which sum up the crucial properties of the past. They are called *history variables*.

Very often, there is a crucial variable, which changes over time, but which is never observed. You may abstract the information regarding the state of that variable. In the influence diagram in Figure 9, you have the two paths of A-nodes and B-nodes. The information relevant for the A-line is the decisions and the even-numbered O-nodes $O2, O4, O6, O8$. This information may be abstracted into the history nodes $Ahist_1$, $Ahist_2$, and $Ahist_3$ (see Figure 18. Note that no-forgetting is not assumed). The same can be done for the B-line.

6.3 Information enhancement

Another approach is *information enhancement* (Jensen and Gatti, 2010). You assume that you will know more in the future, than you actually will. For the influence diagram in Figure 9, rather than assuming future abstracted historical information you may assume that you in the future will know the state of an A-variable and a B-variable. If you assume that when taking decision $D4$ you actually know the state of $A3$ and $B3$, then the relevant variables are $\{A3, B3, O7, O8\}$ (see Figure 19), and you have a much smaller domain for δ_4.

You may also use a closer approximation by assuming knowledge of $B2$ rather than $B3$. In that case δ_4 will have the domain $\{B2, O5, D3, A3, O7, O8\}$.

It should be stressed that information enhancement as well as information abstraction are methods for approximating policies for some of the first decisions and not for the decisions for which you alter the information.

7 Modeling Domains with Multiple Agents

The standard influence diagram framework (as well as the extensions discussed above) only supports the specification of decision problems for a single decision maker who is in complete control of all decisions. For example, in the influence diagram representation of the poker problem (see Figure 8) we have modeled the decisions of player M by assuming that the decisions of opponent O follow a fixed strategy, i.e., O is implemented as an automaton. This assumption is often overly simplistic, and instead we could consider modeling the opponent as an agent that is (also) trying to maximize her own expected utility.

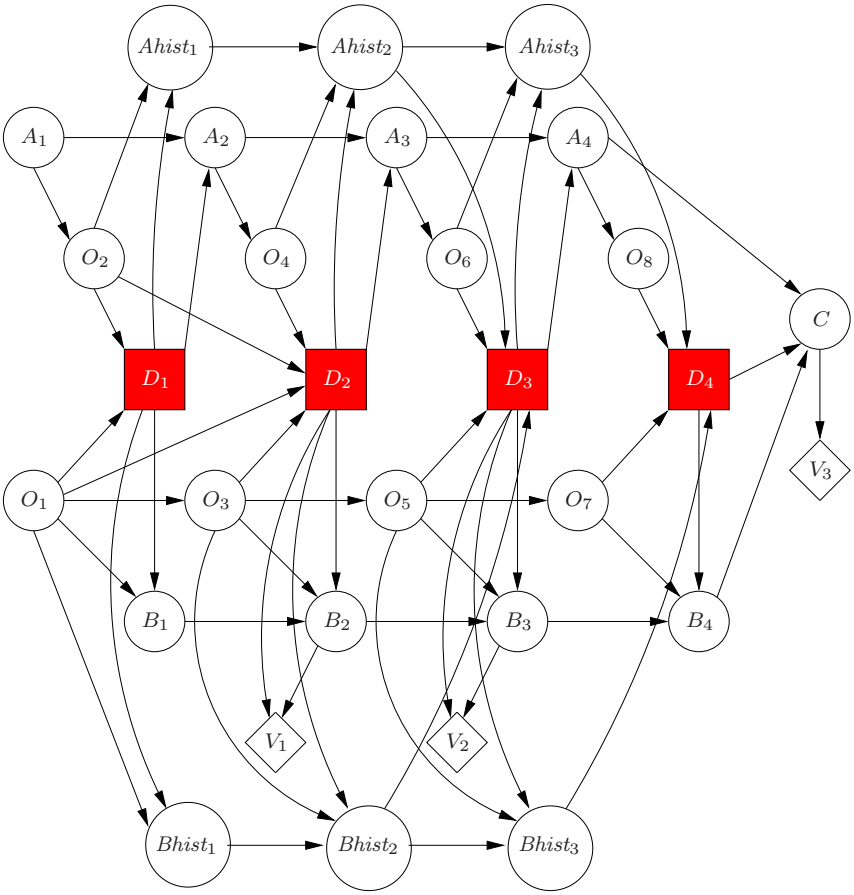

Fig. 18 The model in Figure 9 extended with history variables. No-forgetting is not assumed.

7.1 Multi-agent Influence Diagrams

Traditionally, decision problems involving several decision makers/agents have been modeled by *game trees*, a representation language that can be seen as extending the decision tree framework (Raiffa and Schlaifer, 1961) to multi-agent decision problems. By this measure, game trees suffer from the same complexity problems as decision trees, and this problem has motivated the development of languages for more compact representation. One of the more popular frameworks developed is the *multi-agent influence diagram* (MAID) (Koller and Milch, 2001, 2003), which extends the influence diagram with features for representing non-cooperative games. Specifically, in a MAID, each decision variable is associated with an agent, and the preferences of an agent are represented by a set of utility functions specific for that agent. The total utility for an agent is the sum of the agent's own local utilities.

Figure 20 shows a MAID representation of the poker problem with both agents modeled explicitly. In the model we assume no-forgetting for each of the players,

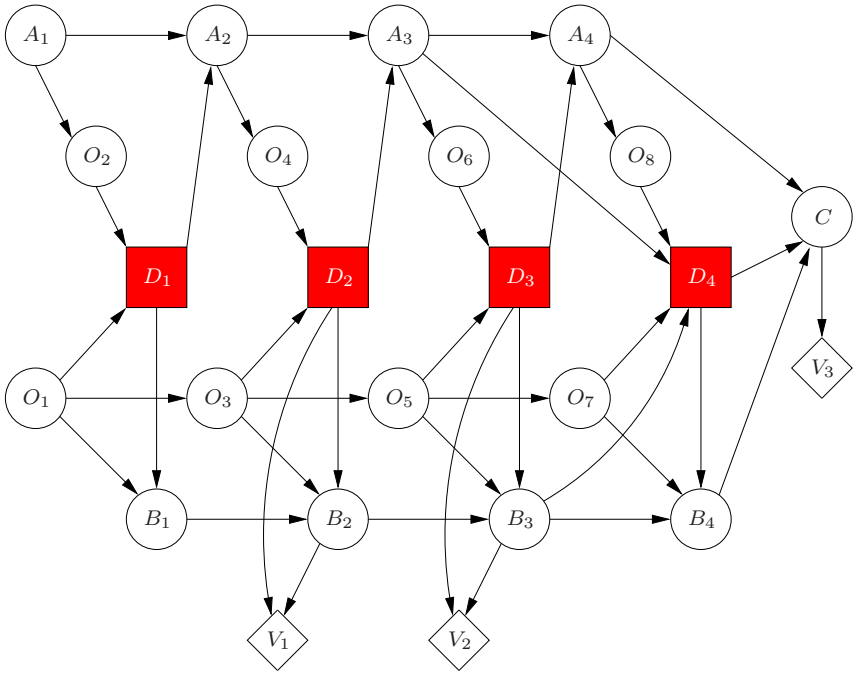

Fig. 19 The model in Figure 9 extended with the links $A_3 \rightarrow D_4$ and $B_3 \rightarrow D_4$.

i.e., the players will remember their own previous decisions and observations. In this example, the utility functions of both players are defined over the same domain and they are therefore represented by a single utility node.

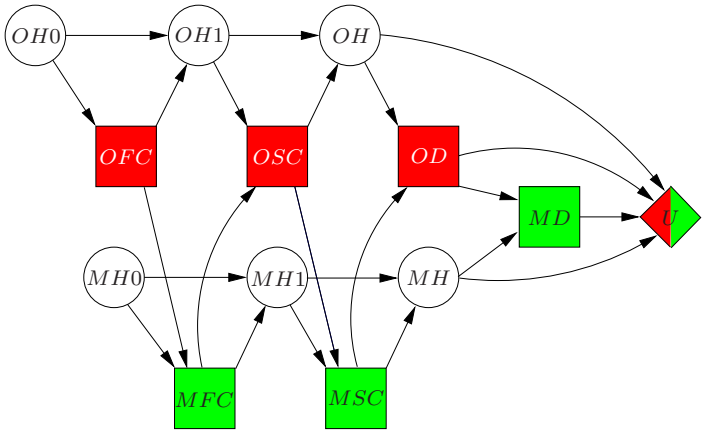

Fig. 20 A multi-agent influence diagram representation of the poker problem.

When analyzing the MAID in Figure 20 we find that the optimal strategy for player M depends on the optimal strategy for player O and vice versa. This means

that we have an infinite regression, where player M models the decision process of player O which includes a model of the decision process of player M and so on. In game theory, a solution to such an infinite regression is found by determining a Nash equilibrium for the decisions. A *Nash equilibrium* is a collection of strategies, one for each agent, with the property that no agent can increase its expected utility by unilaterally deviating from its prescribed strategy. More formally, let $\boldsymbol{\sigma} = \{\sigma_1, \ldots, \sigma_n\}$ be a collection of strategies for n agents. The strategy σ_i is said to be *locally optimal* wrt. $\boldsymbol{\sigma}_{-i} = \boldsymbol{\sigma} \setminus \{\sigma_i\}$ if

$$EU(\boldsymbol{\sigma}) \geq EU(\{\sigma_i'\} \cup \boldsymbol{\sigma}_{-i}),$$

for all σ_i'. Moreover, the collection of strategies $\boldsymbol{\sigma}$ is a Nash equilibrium if for all agents i, σ_i is a locally optimal strategy wrt. $\boldsymbol{\sigma}_{-i}$.

A straightforward approach to finding a Nash equilibrium for a MAID is to first convert it into a game (in either extensive or normal form), and then solve it using existing algorithms from game theory. This approach is analogous to solving an influence diagram by first converting it into a decision tree, and then using backwards induction for solving the tree. It is, however, possible to devise more clever solution procedures that, e.g., exploit the probabilistic independencies encoded in the MAID structure. For example, one may attempt to decompose the general inference task into a collection of smaller sub-tasks, and then find a general solution by combining the solutions to the sub-tasks. The key idea is to first construct a so-called *relevance graph* over the decision nodes (an example is given in Figure 21). In a relevance graph, there is an arc from a decision node D to a decision node D' if the optimal policy for D depends on D'. If there is also an arc from D' to D, then the decision policies for the two decisions are intertwined and an equilibrium should therefore be found. For example, in Figure 21 we have an arc from MFC to OFC, since OFC is observed before MFC and knowledge about the policy for OFC may therefore provide information about O's initial hand $OH0$; the updated belief about $OH0$ may influence the decision at MFC. Similarly, we have an arc from OFC to MFC, since the decision at MFC (and thereby also the policy for MFC) influences the expected utility at OFC.

Given the relevance graph for a MAID, we can identify maximal sets of decision variables that jointly rely on each other (but on no other decision variables) and for which a Nash equilibrium should be computed. These sets of decision variables are also called *strongly connected components*. Syntactically, a maximal set $\mathcal{C}$ of decision variables is a strongly connected component if, for all pairs of decision nodes D and D' in $\mathcal{C}$, there is a directed path from D to D'. In Figure 21 the relevance graph consists of a single strongly connected component that contains all the decision variables. However, imagine that M observes O's initial hand before the first decision (this corresponds to adding an arc from $OH0$ to MFC in Figure 20). In this modified poker problem, the policies for MFC, MSC, and MD do not rely on the policy for OFC, and the resulting relevance graph would therefore contain two strongly connected components: one consisting of only OFC and the other consisting of the remaining decision variables.

The strongly connected components in a relevance graph can always be organized in a directed acyclic graph, which defines a topological ordering of the components. By going in reverse order, we can compute an equilibrium for the last set of decision variables (that does not rely on any other decision variables).

These decision variables can then be substituted by chance variables, and we can then (iteratively) continue to the next set of decision variables. Notice that this procedure resembles the solution procedure for influence diagrams (see Section 4.2) except that we here work with sets of decision variables at each step. In some cases, this decomposition can result in substantial reductions in computational complexity compared to solving the full game tree, but in the worst case all decision variables rely on each other and no reductions can therefore be obtained. The poker problem is one such example: the optimal policy for any of the decision variables (indirectly) relies on the policies of all the other decision variables (see Figure 21).

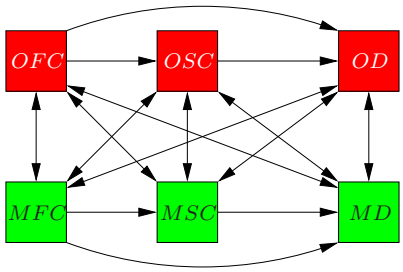

Fig. 21 The relevance graph for the multi-agent influence diagram representation of the poker problem. There is a directed path between all pairs of decisions, which implies that we need to consider all decisions simultaneously when finding a Nash equilibrium.

7.2 Modeling the Decision Making Processes of other Agents

Fundamental to the MAID framework is the assumption that the model of the problem is known to all the agents and that the model captures all relevant aspects required for calculating optimal decision policies for the agents. In particular, it is assumed that the agents are knowledgeable about the decision policies available to the other agents. These assumption will often make the optimal policies interdependent, in which case a solution to the decision problem is a Nash equilibrium. However, in most real-world problems (human) agents do not calculate a Nash equilibrium when solving a decision problem nor do they necessarily rely on the same model for making decisions. Instead, agents may have their own private mental models, which they use when making decisions. These observations have motivated the development of several frameworks that separate the structure of the game/decision problem from the mental models used by the agents when doing reasoning.

7.2.1 Network of Influence Diagrams

The *network of influence diagrams* (NIDs) (Gal and Pfeffer, 2003, 2008) is such a framework, which facilitates the separation of the real-world model from the agents' mental models. For example, it is possible for an agent to have several

models of another agent, using a probability distribution to represent the uncertainty about which model is actually being used by the agent. Moreover, the mental models may themselves contain mental models of other agents, thus supporting a type of recursive modeling (Gmytrasiewicz et al, 1991).

A NID is a rooted directed acyclic graph, in which each node is a MAID. To make a clear distinction between the nodes in a NID and the nodes in a MAID, the nodes in a NID are usually referred to as blocks. The root block of a NID is a MAID representing the real-world from the modeler's point of view. An edge from a block X to a block Y is labeled with $\{A, \mathbf{D}\}$, meaning that in MAID X agent A is assumed to use the mental model Y when making decisions $\mathbf{D}$. As an example, consider again the poker problem, where the MAID in Figure 20 corresponds to the real-world model from M's point of view. Suppose that M believes that O plays according to one of the following models:

- a fixed strategy (encoded in the conditional probability tables associated with the chance nodes OFC, OSC, and OD in Figure 8);
- the MAID in Figure 20, where O in turn believes that M plays according to the model in Figure 8) (i.e., O believes that M believes that O plays a naive strategy and will therefore play a best response to that strategy);
- the MAID in Figure 20 with an infinite regression being the result.

Thus, M has three mental models for O with regression levels 1, 3, and infinity. Using the NID framework, this revised poker problem can be represented as in Figure 22. Agent M's uncertainty about which model O is using for deciding on OFC, OSC, and OD is modeled by a *model selection variable* $\text{Mod}[OFC, OSC, OD]$ having three states corresponding to the three blocks in the NID. Graphically, $\text{Mod}[OFC, OSC, OD]$ is included as a parent of the three decisions OFC, OSC, and OD in Figure 20 (see Figure 23); the significance of this construction becomes apparent in the solution phase. Note that $\text{Mod}[OFC, OSC, OD]$ is an ordinary chance variable that could also be influenced by other variables in the model.

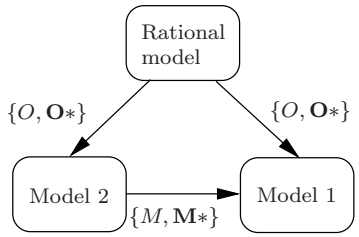

Fig. 22 A NID representation of the poker problem; $\mathbf{O}*$ and $\mathbf{M}*$ are shorthand notations for $\{OFC, OSC, OD\}$ and $\{MFC, MSC, MD\}$. Model 1 and 2 correspond to the models in Figures 8 and 20, respectively, and the rational model corresponds to the model in Figure 23. Player M's uncertainty about which model O uses for deciding on OFC, OSC, and OD is modeled by a chance variable $\text{Mod}[OFC, OSC, OD]$.

When solving a NID we proceed bottom-up using a topological ordering of the blocks in the DAG. The blocks in the leafs are simply MAIDs, which can be solved using the algorithm outlined above. In case the model only contains decision variables for a single agent (as in Model 1 in Figure 22) a solution can be found

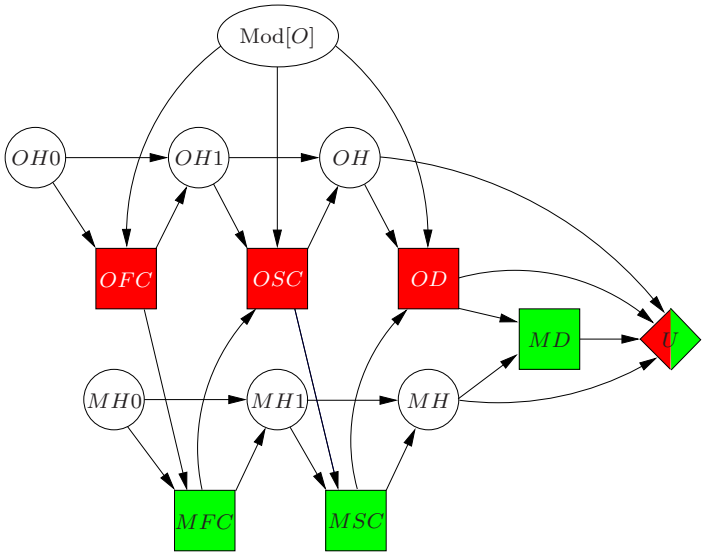

Fig. 23 The top-level rational block for the NID shown in Figure 22. M's uncertainty about which model O uses is captured by the model selection node Mod[O], which corresponds to a standard chance variable with three states representing the three possible models.

using a standard solution algorithm for influence diagrams. When all the children of an internal block X have been solved, the calculated policies are incorporated into X. This involves representing the policies by their chance variable representations and using the model selecting variables (e.g. Mod[OFC, OSC, OD]) for making a conditional specification of which policies to use. Block X can then be solved and the calculated decision policies are passed on to its parents.

8 Other Issues

In the sections above we have mainly focused on modeling aspects as well as general considerations about inference. There are, however, several other research areas in connection to probabilistic decision graphs that we have not touched upon and for which a more thorough discussion will be outside the scope of the present paper. Below we will give a brief overview on some of the developments with emphasis on modeling scope, inference, and domain analysis.

So far we have assumed that all variables are discrete, but in many real-world domains we often find a mixture of discrete and continuous variables. A straightforward approach for modeling and solving these domains is simply to discretize the continuous variables, and apply one of the standard frameworks discussed above. Alternatively, there have also been several attempts at extending the influence diagram and the solution algorithms to allow for continuous variables in the representation. For example, Shachter and Kenley (1989) propose the *Gaussian influence diagram*, where all variables (including the decision variables) are continuous and the chance variables are assigned linear Gaussian distributions. More recently, Madsen and Jensen (2005) and Madsen (2008) have proposed methods

for solving *conditional linear quadratic Gaussian* (CLQG) IDs containing both discrete and continuous variables. In a CLQG ID, the continuous variables follow conditional linear Gaussian distributions and the utility functions are quadratic functions over any continuous parents. Furthermore, discrete children of continuous parents are not allowed and the temporal order defined by the CLQG ID should specify that all discrete variables have been observed or decided upon before any of the continuous variables. The latter requirement can be weakened, but this requires a structural analysis of the dependence properties specified by the graph structure. In combination, these requirements ensure that an exact solution can be found and, in particular, the (negative) quadratic utility functions ensure that a unique optimal decision can be found for the decision variables without having to solve a complex optimization problem.

The representational restrictions imposed by the solution methods for CLQG IDs do, however, limit the applicability of these models and have motivated the development of alternative modeling frameworks. One recently proposed model is based on the *mixture of truncated exponentials* (MTEs) framework, which was originally specified for Bayesian networks (Moral et al, 2001). In an MTE-based Bayesian network, discrete and continuous variables can be treated in a uniform fashion and exact inference can be performed using the Shafer-Shenoy architecture (Shafer and Shenoy, 1990). Cobb and Shenoy (2004, 2008) extended the MTE framework to influence diagrams with the only requirement that the decision variables should be discrete. This restriction was later lifted by Cobb (2006). More recently, Li and Shenoy (2010) proposed a general architecture (exemplified using mixtures of polynomials (Shenoy and West, 2009)) for solving hybrid influence diagrams with deterministic functions assigned to (some of) the chance variables.

Algorithms for solving decision graphs, and influence diagrams in particular, have received much attention. In general, solving an influence diagram is NP-hard (Cooper, 1990) but many real-life problems have been shown to have feasible solutions. Exact algorithms for solving influence diagrams range from algorithms working directly on the influence diagram structure (Shachter, 1986) to algorithms that rely on a secondary representation of the model.[2] Examples of the latter include search-based algorithms that operate on a decision tree representation of the model (Yuan and Wu, 2010; Yuan et al, 2010), message-passing algorithms that rely on a junction tree representation of the model (Jensen et al, 1994; Madsen and Jensen, 1999; Madsen and Nilsson, 2001), and algorithms that transform the influence diagram into a so-called decision circuit (Bhattacharjya and Shachter, 2007; Shachter and Bhattacharjya, 2010), the analogue to arithmetic circuits for Bayesian networks (Darwiche, 2003). The computational difficulties involved in solving influence diagrams have also sparked the development of several approximate algorithms. For example, Horsch and Poole (1998) and Dechter (2000) have proposed anytime algorithms for solving influence diagrams, and sampling-based methods have been described in e.g. (Virto et al, 2002; Charnes and Shenoy, 2004; Cano et al, 2006; Garcia-Sanchez and Druzdzel, 2007).

Given an influence diagram representation of a decision problem, we can not only use the model to find an optimal strategy for the decisions involved, but we may also use the model to perform a more extensive analysis of the decision

[2] It is worth noting that solving an influence diagram is closely related to performing inference in a Bayesian network (Cooper, 1988; Shachter and Peot, 1992; Zhang, 1998).

problem. For example, in Section 3.1 we mentioned algorithms for determining the required past of a decision variable based on the structure of the model; similar algorithms also exist for determining the relevant future of a decision (Nielsen, 2001). For analyzing the numerical aspects of a model, algorithms have been developed for performing *sensitivity analyses*, i.e., determining how sensitive the maximum expected utility or the optimal strategy is to variations of one or more of the (numerical) parameters in the model (Felli and Hazen, 1999; Bielza et al, 2000; Nielsen and Jensen, 2003; Bhattacharjya and Shachter, 2008). A related type of analysis is *value of information*: should we look for more information before deciding on a particular decision? When several different information sources are available, the general problem becomes intractable and one typically resorts to approximate algorithms (Dittmer and Jensen, 1997; Shachter, 1999; Liao and Ji, 2008)

9 Computer systems

There are several academic as well as commercial systems for editing and running Bayesian networks. You can find a list on `directory.google.com/Top/Computers/Artificial_Intelligence/Belief_Networks/Software`.

Some of the systems also process influence diagrams. For example, the commercial systems Analytica (`www.lumina.com`), Bayesia (`www.bayesia.com`), and Netica (`www.norsys.com`) process influence diagram specifications with no-forgetting, and Hugin (`www.hugin.com`) is based on LIMIDs.

The academic system Bayesian network toolbox (`www.cs.ubc.ca/~murphyk/Software/BNT/bnt.html`), Java Bayes (`www.cs.cmu.edu/~javabayes/Home`), Genie (`genie.sis.pitt.edu/`), and MSBNx (`http://research.microsoft.com/en-us/um/redmond/groups/adapt/msbnx`) process influence diagrams, and the Elvira system `leo.ugr.es/elvira` also processes UIDs.

References

Ahlmann-Ohlsen KS, Jensen FV, Nielsen TD, Pedersen O, Vomlelova M (2009) A comparsion of two approaches for solving unconstrained influence diagrams. International Journal of Approximate Reasoning 50:153–173

Bhattacharjya D, Shachter RD (2007) Evaluating influence diagrams with decision circuits. In: Proceedings of the Twentythird Conference on Uncertainty in Artificial Intelligence (UAI), pp 9–16

Bhattacharjya D, Shachter RD (2008) Sensitivity analysis in decision circuits. In: Proceedings of the Twentyfourth Conference on Uncertainty in Artificial Intelligence (UAI), pp 34–42

Bielza C, Ríos-Insua S, Gómez M, del Pozo JAF (2000) Sensitivity analysis in ictneo. In: Lecture Notes in Statistics, Robust Bayesian Analysis, vol 152, Springer, pp 317–334

Cano A, Gómez M, Moral S (2006) A forward-backward Monte Carlo method for solving influence diagrams. International Journal of Approximate Reasoning 42(1–2):119–135

Charnes JM, Shenoy PP (2004) Multistage monte carlo method for solving influence diagrams using local computation. Management Science 50:405–418

Cobb B (2006) Continuous decision MTE influence diagrams. In: Proceedings of the Third European Workshop on Probabilistic Graphical Models, pp 67–74

Cobb BR, Shenoy PP (2004) Hybrid influence diagrams using mixtures of truncated exponentials. In: Proceedings of the 20th conference on Uncertainty in artificial intelligence, AUAI Press, Arlington, Virginia, United States, Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence (UAI), pp 85–93, URL http://portal.acm.org/citation.cfm?id=1036843.1036854

Cobb BR, Shenoy PP (2008) Decision making with hybrid influence diagrams using mixtures of truncated exponentials. European Journal of Operational Research 186(1):261–275, DOI DOI:10.1016/j.ejor.2007.01.036

Cooper GF (1988) A method for using belief networks as influence diagrams. In: Proceedings of the Fourth Conference on Uncertainty in Artificial Intelligence, pp 55–63

Cooper GF (1990) The computational complexity of probabilistic inference using Bayesian belief networks. Artificial Intelligence 42(2–3):393–405

Darwiche A (2003) A differential approach to inference in Bayesian networks. Journal of the ACM 50(3):280–305

Darwiche A (2009) Modeling and Reasoning with Bayesian Networks. Cambridge University Press

Dechter R (2000) An anytime approximation for optimizing policies under uncertainty. In: In Workshop on Decision Theoretic Planning, at the 5th International Conference on Artificial Intelligence Planning Systems (AIPS-2000)

Dittmer SL, Jensen FV (1997) Myopic value of information in influence diagrams. In: Geiger D, Shenoy PP (eds) Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, pp 142–149

Felli JC, Hazen GB (1999) Sensitivity analysis and the expected value of perfect information. Medical Decision Making 18:95–109

Gal Y, Pfeffer A (2003) A language for modeling agents' decision making processes in games. In: Proceedings of the Second International Joint Conference on Autonomous agents and multiagent systems, ACM Press, pp 265 – 272

Gal Y, Pfeffer A (2008) Networks of influence diagrams: a formalism for representing agent's beliefs and decision making processes. Journal of Artificial Intelligence Research 33:109–147

Garcia-Sanchez D, Druzdzel M (2007) An efficient exhaustive anytime sampling algorithm for influence diagrams. In: Lucas P, Gmez J, Salmern A (eds) Advances in Probabilistic Graphical Models, Studies in Fuzziness and Soft Computing, vol 214, Springer Berlin / Heidelberg, pp 255–273

Gmytrasiewicz PJ, Durfee EH, Wehe DK (1991) A decision-theoretic approach to coordinating multiagent interactions. In: Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, pp 62–68

Horsch MC, Poole D (1998) An anytime algorithm for decision making under uncertainty. In: Cooper GF, Moral S (eds) Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI), Morgan Kaufmann Publishers, pp 246–255

Howard RA, Matheson JE (1981) Influence diagrams. In: Howard RA, Matheson JE (eds) The Principles and Applications of Decision Analysis, vol 2, Strategic Decision Group, chap 37, pp 721–762

Jensen F, Jensen FV, Dittmer SL (1994) From influence diagrams to junction trees. In: de Mantaras RL, Poole D (eds) Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, pp 367–373

Jensen FV, Gatti E (2010) Information enhancement for approximate representation of optimal strategies for influence diagrams. In: Proceedings of the Fifth European Workshop on Probabilistic Graphical Models, pp 102–9

Jensen FV, Nielsen TD (2007) Bayesian Networks and Decision Graphs, 2nd edn. Springer-Verlag New York, ISBN: 0-387-68281-3

Jensen FV, Vomlelova M (2002) Unconstrained influence diagrams. In: Darwiche A, Friedman N (eds) Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, pp 234–241

Jensen FV, Nielsen TD, Shenoy PP (2006) Sequential influence diagrams: a unified framework. International Journal of Approximate Reasoning 42:101–118

Kjaerulff UB, Madsen AL (2008) Bayesian Networks and Influence Diagrams, 1st edn. Springer-Verlag New York, ISBN: 978-0-387-74100-0

Koller D, Friedman N (2009) Probabilistic Graphical Models: Principles and Techniques. MIT Press

Koller D, Milch B (2001) Multi-agent influence diagrams for representing and solving games. In: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, pp 1027–1036

Koller D, Milch B (2003) Multi-agent influence diagrams for representing and solving games. Games and economic behavior 45(1):181–221

Korb KB, Nicholson AE (2004) Bayesian Artificial Intelligence, 1st edn. Chapman and Hall, ISBN: 1-58488-387-1

Lauritzen SL, Nilsson D (2001) Representing and solving decision problems with limited information. Management Science 47(9):1235–1251

Li Y, Shenoy PP (2010) Solving hybrid influence diagrams with deterministic variables. In: Proceedings of the Twentysixth Conference on Uncertainty in Artificial Intelligence (UAI), pp 322–331

Liao W, Ji Q (2008) Efficient non-myopic value-of-information computation for influence diagrams. International Journal of Approximate Reasoning 49:436–450

Luque M, Díez FJ (2010) Variable elimination for influence diagrams with super value nodes. International Journal of Approximate Reasoning 51:615–631

Madsen AL (2008) New methods for marginalization in lazy propagation. In: Jaeger M, Nielsen TD (eds) Proceedings of the Fourth European Workshop on Probabilistic Graphical Models, pp 193–200

Madsen AL, Jensen F (2005) Solving linear-quadratic conditional Gaussian influence diagrams. International Journal of Approximate Reasoning 38:263–282, DOI http://dx.doi.org/10.1016/j.ijar.2004.05.006, URL `http://dx.doi.org/10.1016/j.ijar.2004.05.006`

Madsen AL, Jensen FV (1999) Lazy evaluation of symmetric Bayesian decision problems. In: Laskey KB, Prade H (eds) Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, pp 382–390

Madsen AL, Nilsson D (2001) Solving influence diagrams using HUGIN, Shafer-Shenoy, and lazy propagation. In: Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI), Morgan Kaufmann Publishers,

pp 337–345
Moral S, Rumí R, Salmerón A (2001) Mixtures of truncated exponentials in hybrid Bayesian networks. In: Proceedings of the Sixth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, Lecture Notes in Artificial Intelligence, vol 2143, Springer-Verlag, Berlin, Germany, pp 145–167
von Neumann J, Morgenstern O (1944) Theory of Games and Economic Behavior, 1st edn. John Wiley & Sons, New York
Nielsen TD (2001) Decomposition of influence diagrams. In: Benferhat S, Besnard P (eds) Proceedings of the Sixth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, Springer-Verlag, no. 2143 in Lecture Notes in Artificial Intelligence, pp 144–155
Nielsen TD, Jensen FV (1999) Well-defined decision scenarios. In: Laskey KB, Prade H (eds) Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI), Morgan Kaufmann Publishers, pp 502–511
Nielsen TD, Jensen FV (2003) Sensitivity analysis in influence diagrams. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 33(2):223–234
Nilsson D, Lauritzen SL (2000) Evaluating influence diagrams using LIMIDs. In: Boutilier C, Goldszmidt M (eds) Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, pp 436–345
Pearl J (1988) Probabilistic Reasoning in Intelligent Systems. Representation and Reasoning, Morgan Kaufmann Publishers, San Mateo California, iSBN 0-934613-73-7
Raiffa H, Schlaifer R (1961) Applied Statistical Decision Theory. MIT press, Cambridge
Shachter RD (1986) Evaluating influence diagrams. Operations Research 34(6):871–882
Shachter RD (1999) Efficient value of information computation. In: Laskey KB, Prade H (eds) Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, pp 594–601
Shachter RD, Bhattacharjya D (2010) Dynamic programming in influence diagrams with decision circuits. In: Proceedings of the Twentysixth Conference on Uncertainty in Artificial Intelligence (UAI), pp 509–516
Shachter RD, Kenley CR (1989) Gaussian influence diagrams. Management Science 35(5):527–550
Shachter RD, Peot MA (1992) Decision making using probabilistic inference methods. In: Dubois D, Wellman MP, D'Ambrosio B, Smets P (eds) Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, pp 276–283
Shafer GR, Shenoy PP (1990) Probability Propagation. Annals of Mathematics and Artificial Intelligence 2:327–352
Shenoy PP (1992) Valuation-based systems for Bayesian decision analysis. Operations Research 40(3):463–484
Shenoy PP (2000) Valuation network representation and solution of asymmetric decision problems. European Journal of Operations Research 121(3):579–608
Shenoy PP, West JC (2009) Mixtures of polyno- mials in hybrid bayesian networks with determin- istic variables. In: Proceedings of the Eighth Workshop

on Uncertainty Processing, pp 202–212
Virto MA, Martín J, Insua DR, Moreno-Díaz A (2002) Approximate solutions of complex influence diagrams through mcmc methods. In: Proceedings of the First European Workshop on Probabilistic Graphical Models
Yuan C, Wu X (2010) Solving influence diagrams using heuristic search. In: Proceedings of the Eleventh International Symposium on Artificial Intelligence and Mathematics
Yuan C, Wu X, Hansen E (2010) Solving multistage influence diagrams using branch-and-bound search. In: Proceedings of the Twentysixth Conference on Uncertainty in Artificial Intelligence (UAI), Catalina Island, CA
Zhang NL (1998) Probabilistic Inference in Influence Diagrams. In: Cooper GF, Moral S (eds) Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, pp 514–522